



1F2230 Jaringan Komputer Network Security

Robithoh Annur
Andreas Bara Timur
Monterico Andrian
Prof. Nana Rachmana



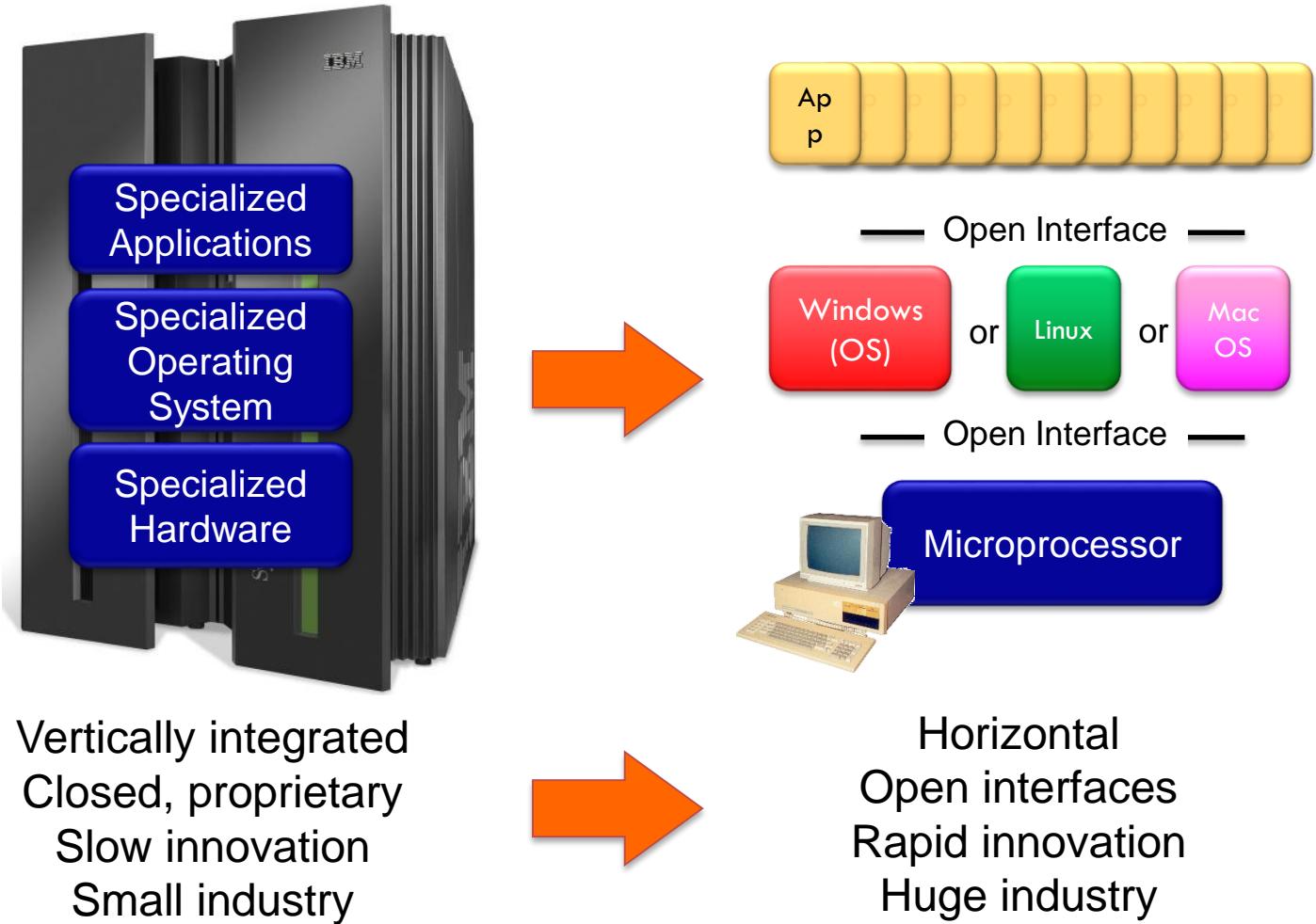
Outline

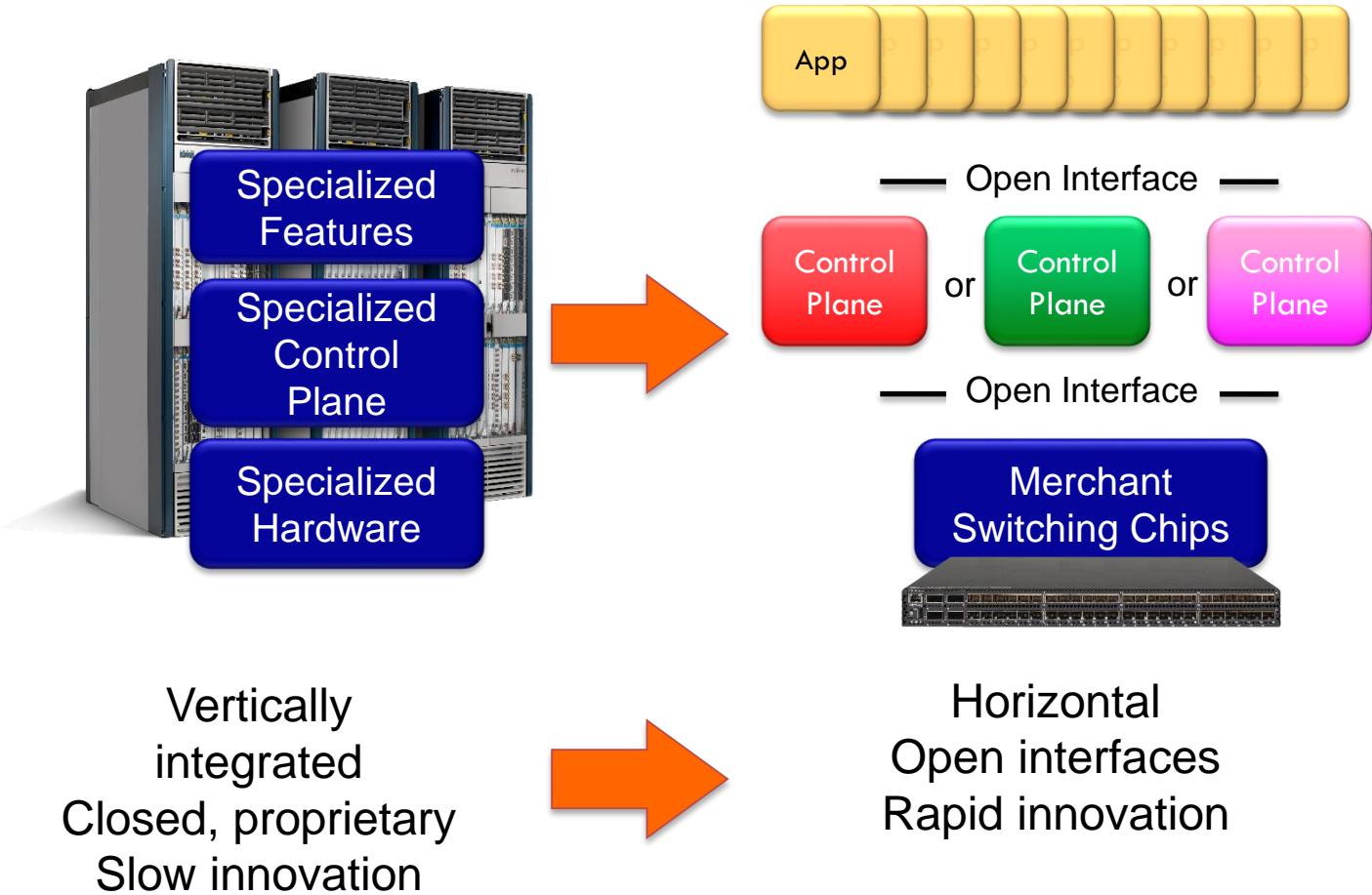
1. What are Software Defined Networks?

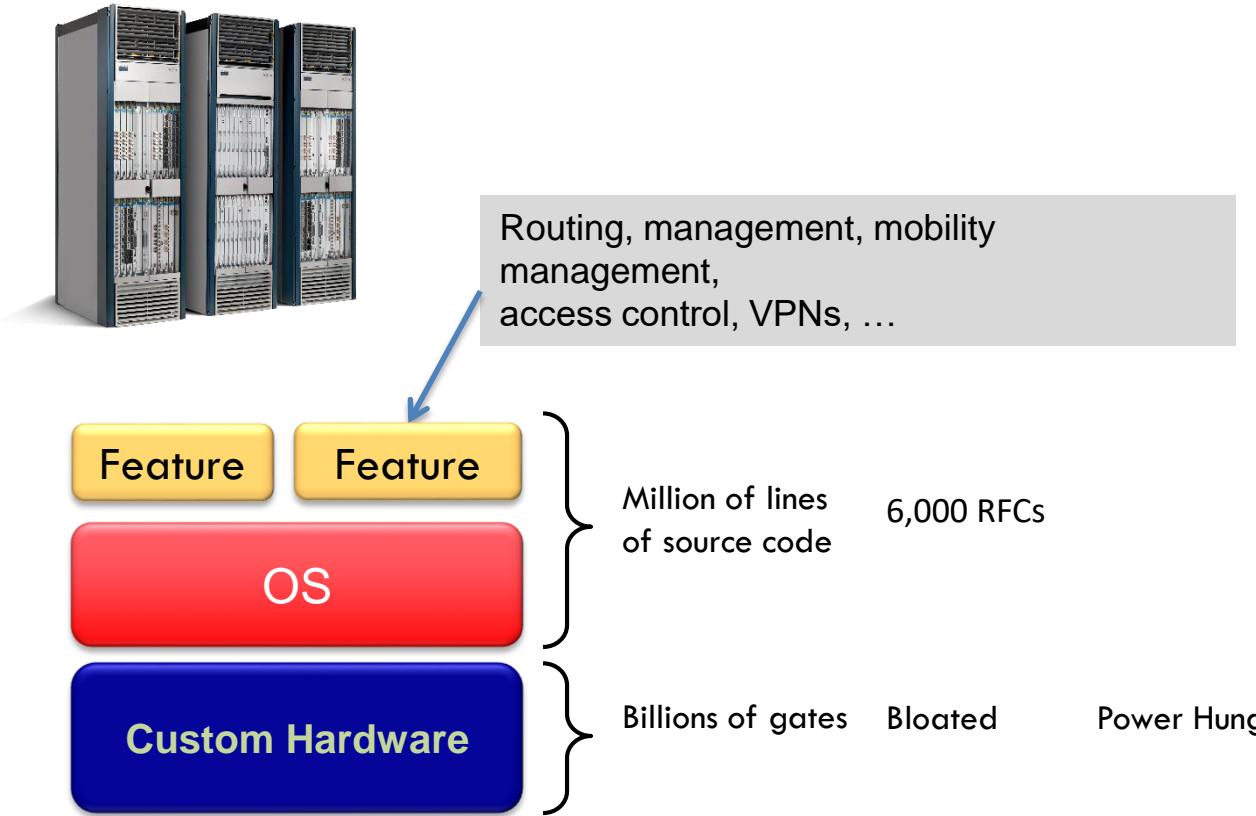
2. Why SDN?

3. The Consequences

- For industry
- For research
- For standards and protocols

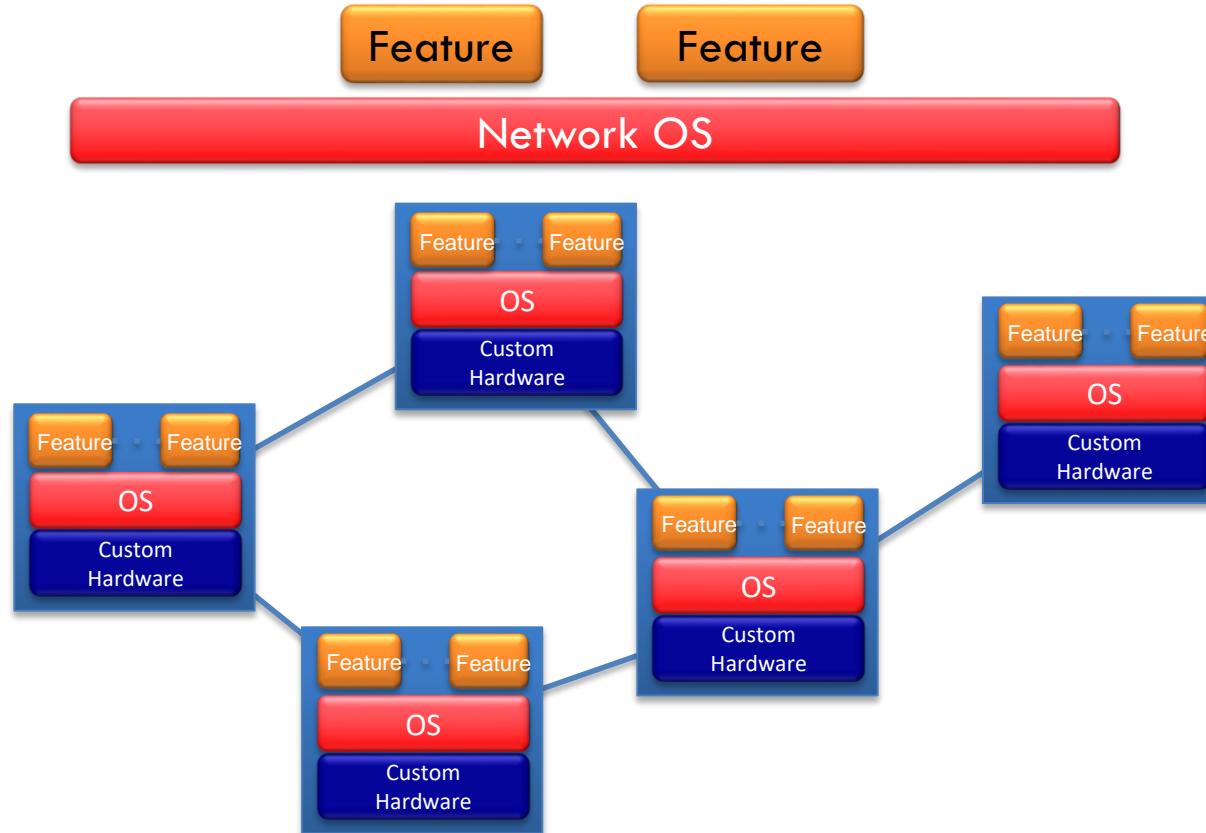






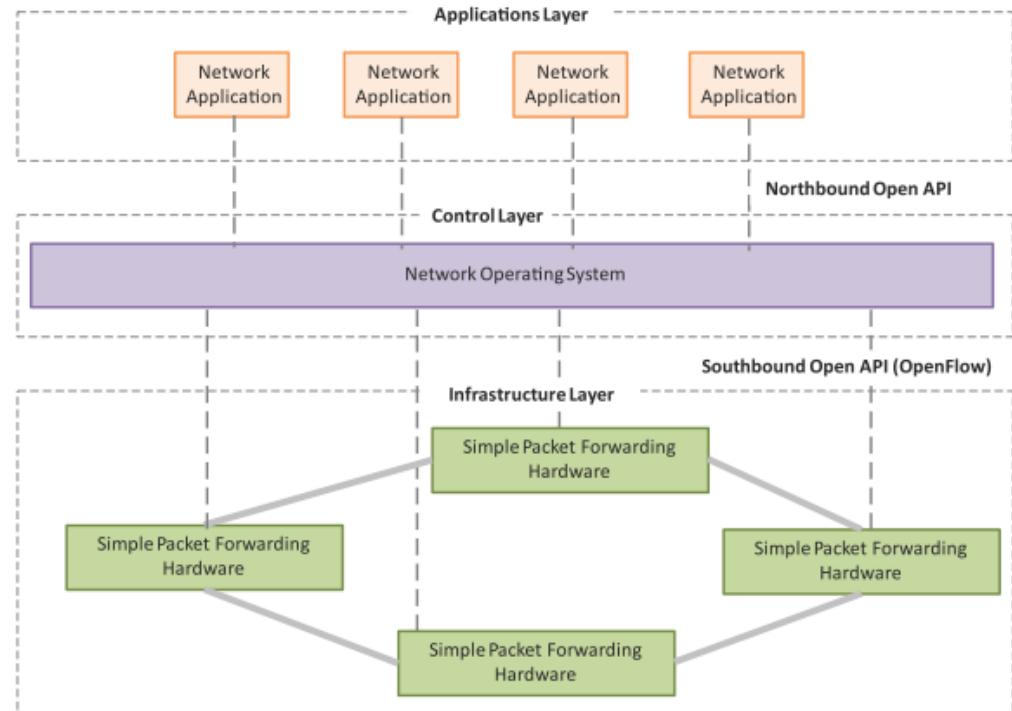
- Vertically integrated, complex, closed, proprietary
- Networking industry with “mainframe” mind-set

The network is changing



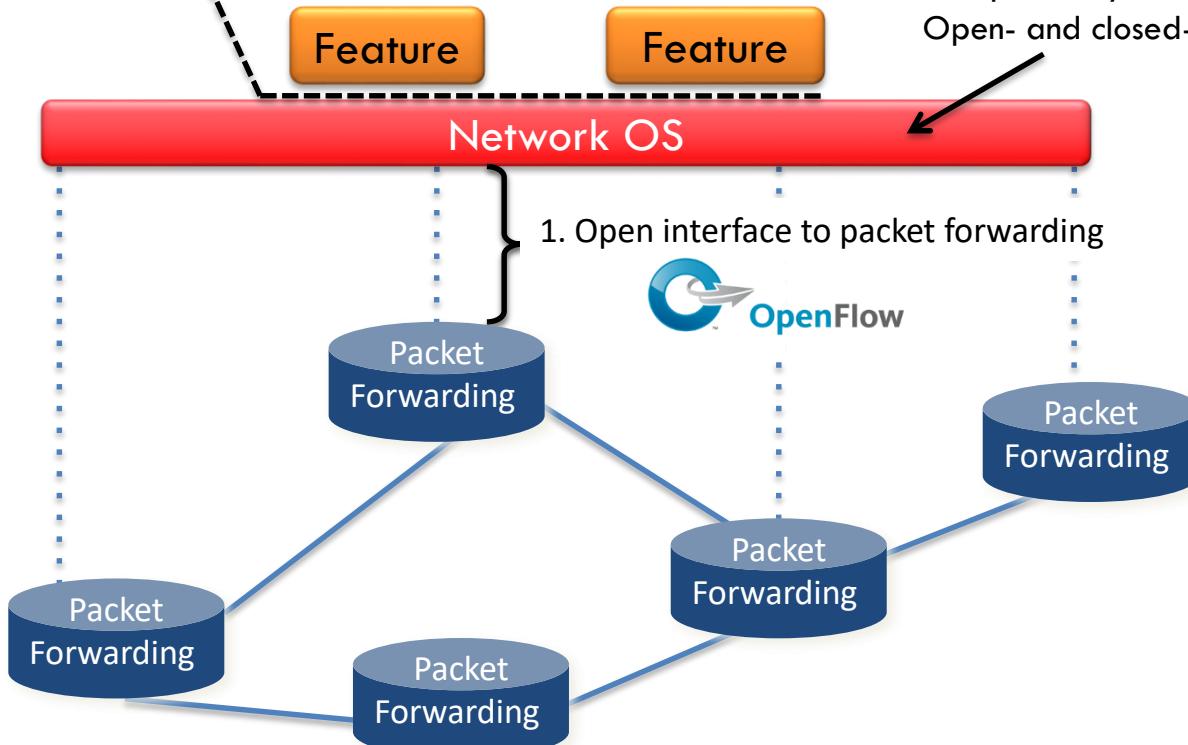
SDN

- Software-Defined Networking (SDN) is a networking architecture that separates the control plane from the data plane and centralizes the network controller.
- Software-based SDN controllers maintain a unified view of the network and make configuration, management and provisioning simpler.
- The underlying infrastructure in SDN uses simple packet forwarding hardware as opposed to specialized hardware in conventional networks.



Software Defined Network (SDN)

3. Consistent, up-to-date global network view





Key Elements of SDN

- Centralized Network Controller
 - With decoupled control and data planes and a centralized network controller, the network administrators can rapidly configure the network.
- Programmable Open APIs
 - SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface).
- Standard Communication Interface (OpenFlow)
 - SDN architecture uses a standard communication interface between the control and infrastructure layers (Southbound interface).
 - OpenFlow, which is defined by the Open Networking Foundation (ONF), is the broadly accepted SDN protocol for the Southbound interface.

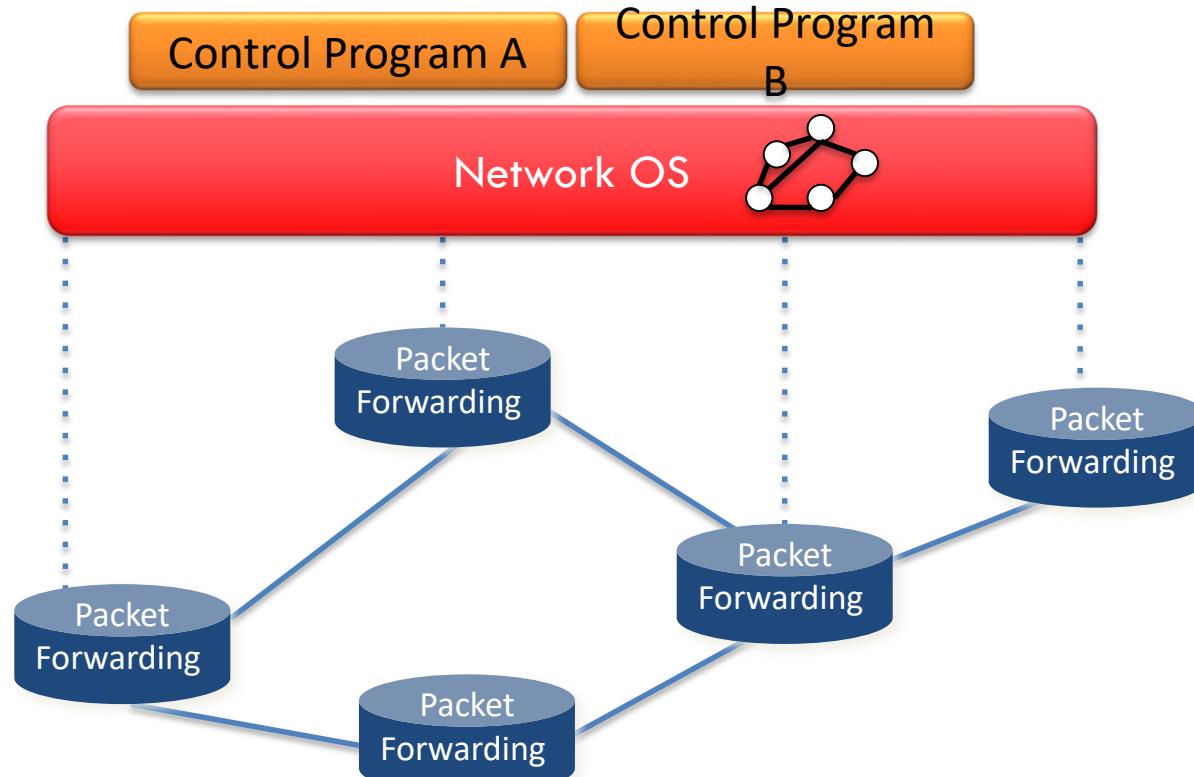
Network OS: distributed system that creates a consistent, up-to-date network view

- Runs on servers (controllers) in the network
- NOX, ONIX, Trema, Beacon, Maestro, ... + more

Uses forwarding abstraction to:

- Get state information **from** forwarding elements
- Give control directives **to** forwarding elements

Software Defined Network (SDN)





Control Program

- Control program operates on view of network
 - **Input:** global network view (graph/database)
 - **Output:** configuration of each network device
- Control program is not a distributed system
 - Abstraction hides details of distributed state

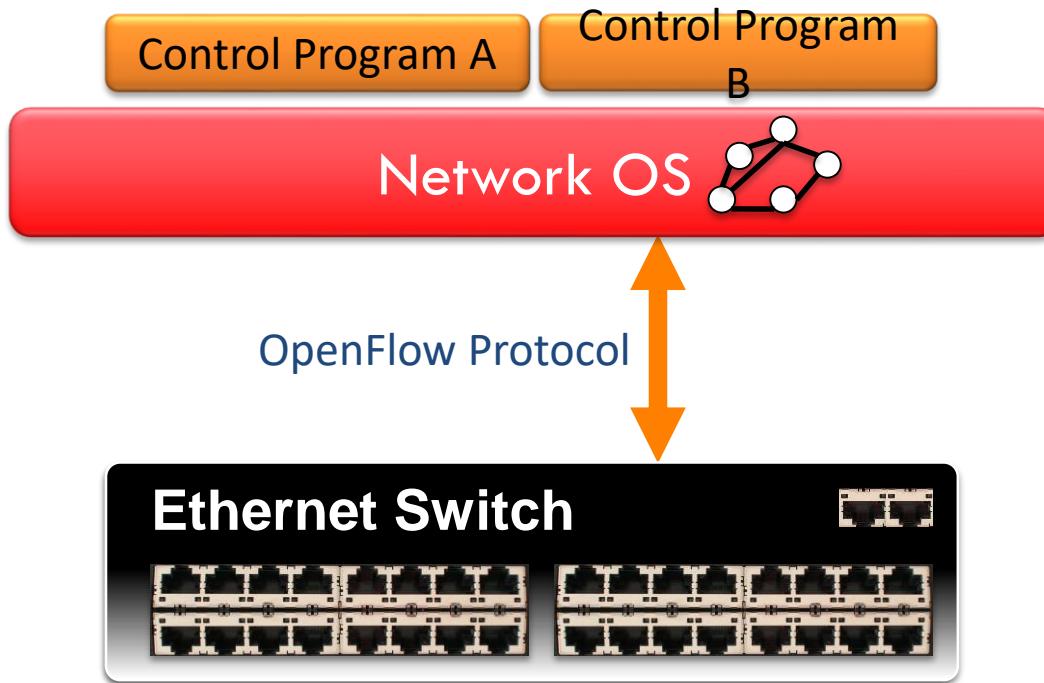


Forwarding Abstraction

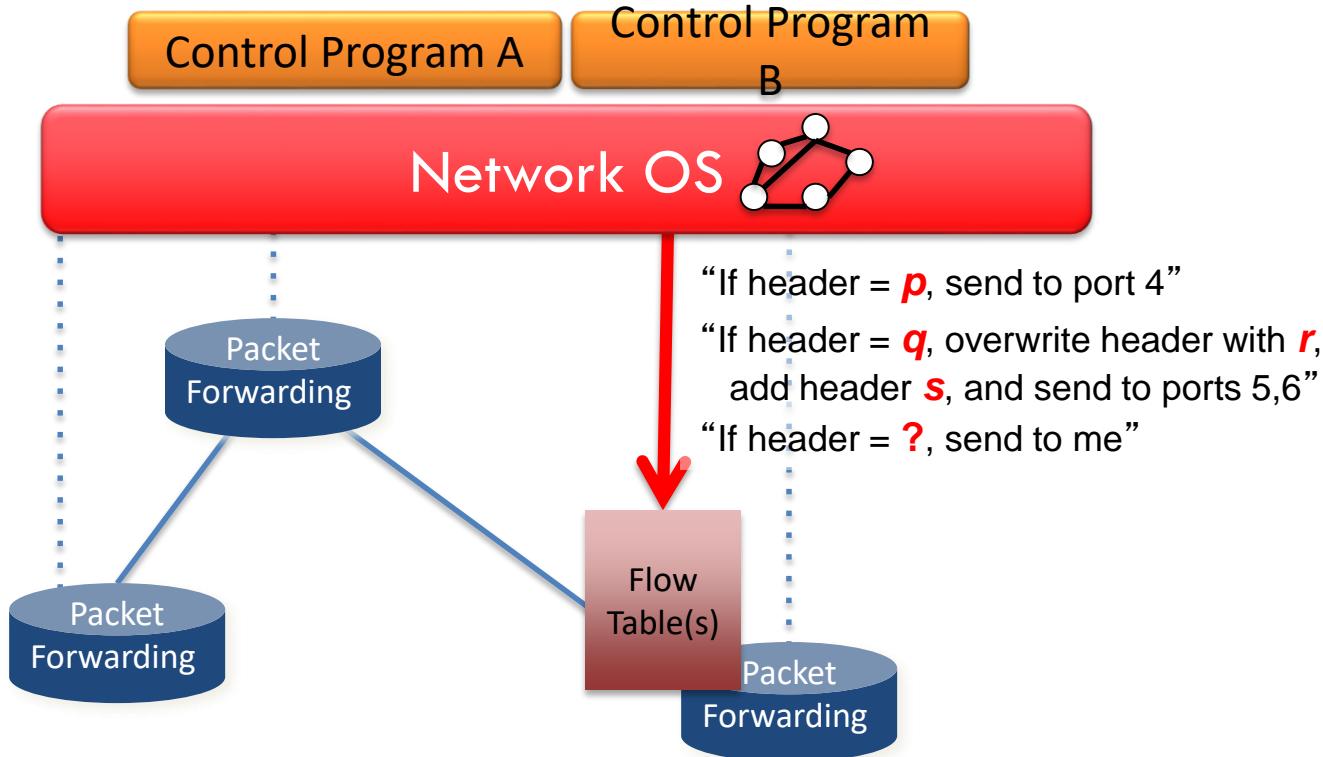
Purpose: Abstract away forwarding hardware

- Flexible
 - Behavior specified by control plane
 - Built from basic set of forwarding primitives
- Minimal
 - Streamlined for speed and low-power
 - Control program not vendor-specific
- OpenFlow is an example of such an abstraction

OpenFlow Basics



OpenFlow Basics





Plumbing Primitives

- Primitive is $\langle \text{Match}, \text{Action} \rangle$
- **Match** arbitrary bits in headers:



- Match on any header, or new header
- Allows any flow granularity
- **Action**
 - Forward to port(s), drop, send to controller
 - Overwrite header with mask, push or pop
 - Forward at specific bit-rate



General Forwarding Abstraction

Small set of primitives
“Forwarding instruction set”

Protocol independent
Backward compatible

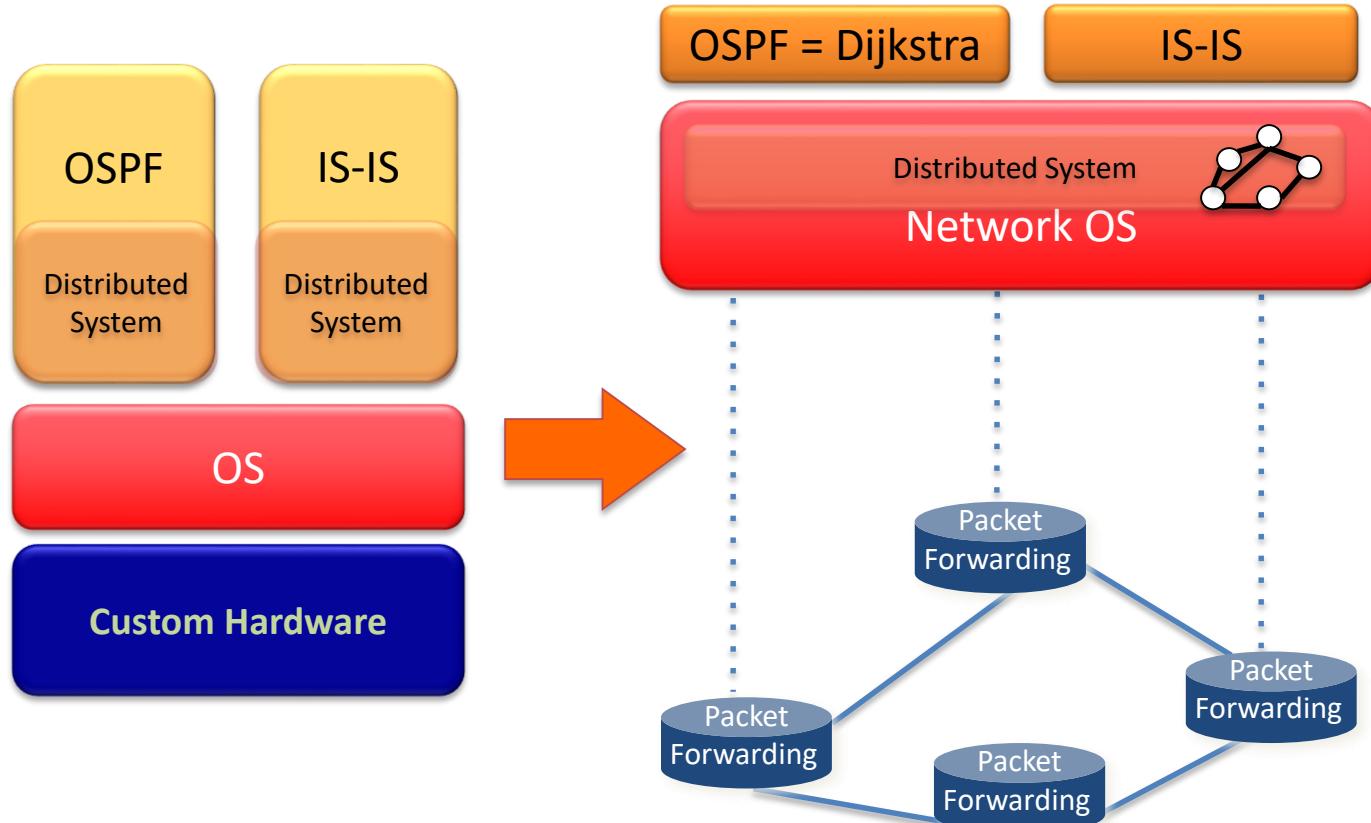
Switches, routers, WiFi APs,
basestations, TDM/WDM



Example 1: OSPF and Dijkstra

- OSPF
 - RFC 2328: **245 pages**
- Distributed System
 - Builds consistent, up-to-date map of the network:
101 pages
- Dijkstra's Algorithm
 - Operates on map: **4 pages**

Example



1. What are Software Defined Networks?

2. Why SDN?

3. The Consequences

- For industry
- For research
- For standards and protocols



Great talk by Scott Shenker

<http://www.youtube.com/watch?v=WVs7Pc99S7w>

(Story summarized here)



Networking

- Networking is
 - “Intellectually Weak”
 - behind other fields
 - about the mastery of complexity

Good abstractions tame complexity

- Interfaces are instances of those abstractions

No abstraction => increasing complexity

- We are now at the complexity limit



By comparison: Programming

- Machine languages: no abstractions
 - Had to deal with low-level details
- Higher-level languages: OS and other abstractions
 - File system, virtual memory, abstract data types,
 - ...
- Modern languages: even more abstractions
 - Object orientation, garbage collection,...



Programming Analogy

- What if programmers had to:
 - Specify where each bit was stored
 - Explicitly deal with internal communication errors
 - Within a programming language with limited expressibility
- Programmers would redefine problem by:
 - Defining higher level abstractions for memory
 - Building on reliable communication primitives
 - Using a more general language



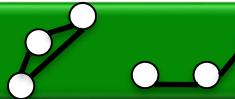
Specification Abstraction

- Network OS eases implementation
 - E.g., Helps manage distributed state
- Next step is to ease specification
 - E.g., How do you specify what the system should do?
- Key goals
 - Provide abstract view of network map
 - Control program operates on abstract view
 - Develop means to simplify specification

Software Defined Network (SDN)

Abstract Network View

Virtualization



Global Network View

Network OS



Packet
Forwarding

Packet
Forwarding

Packet
Forwarding

Consequence:

Work on Network Programming Languages Pyretic, Frenetic etc.

1. What are Software Defined Networks?

2. Why SDN?

3. The Consequences

- For industry
- For research
- For standards and protocols



SDN in development

Domains

- Data centers
- Enterprise/campus
- Cellular backhaul
- Enterprise WiFi
- WANs

Products

- Switches, routers:
About 15 vendors
- Software: About 6
vendors and
startups

New startups (6 so far). Lots of hiring in networking.



Cellular industry

- Recently made transition to IP
- Billions of mobile users
- Need to securely extract payments and hold users accountable
- IP is bad at both, yet hard to change

SDN enables industry to customize their network

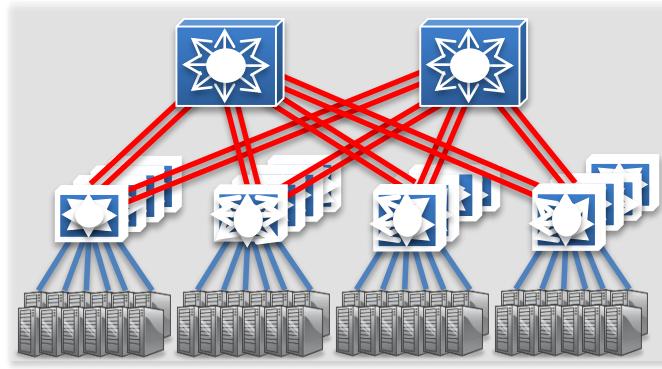


Telco Operators

- Global IP traffic growing 40-50% per year
- End-customer monthly bill remains unchanged
- Therefore, CAPEX and OPEX need to reduce 40-50% per Gb/s per year
- But in practice, reduces by ~20% per year

SDN enables industry to reduce OPEX and CAPEX
...and to create new differentiating services

Example: New Data Center



Cost

200,000 servers

Fanout of 20 → 10,000 switches

\$5k vendor switch = \$50M

\$1k commodity switch = \$10M

Savings in 10 data centers = **\$400M**

Control

More flexible control

Tailor network for services

Quickly improve and innovate



Consequences for research

Ease of trying new ideas

- Existing tools: NOX, Beacon, switches, Mininet
- More rapid technology transfer
- GENI, Ofelia and many more

A stronger foundation to build upon

- Provable properties of forwarding
- New languages and specification tools



Consequences for standards

Standards will define the interfaces

The role of standards will change:

- Network owners will define network behavior
- Features will be adopted without standards

Programming world

- Good software is adopted, not standardized



Summary

- Networks becoming
 - More programmatic
 - Defined by owners and operators, not vendors
 - Faster changing, to meet operator needs
 - Lower opex, capex and power
- Abstractions
 - Will shield programmers from complexity
 - Make behavior more provable
 - Will take us places we can't yet imagine